BUNDESREPUBLIK DEUTSCHLAND

Prioritätsbescheinigung über die Einreichung einer Patentanmeldung

Aktenzeichen: 102 50 638.8

Anmeldetag: 30. Oktober 2002

Anmelder/Inhaber: Siemens Aktiengesellschaft, München/DE

Bezeichnung: Strukturierung, Speicherung und Verarbeitung von Daten gemäß einem generischen Objektmodul

IPC: G 06 F 9/54

Die angehefteten Stücke sind eine richtige und genaue Wiedergabe der ursprünglichen Unterlagen dieser Patentanmeldung.

München, den 30. Oktober 2003
Deutsches Patent- und Markenamt
Der Präsident
Im Auftrag

A 9161
02/00
EDV-L

Beschreibung

Strukturierung, Speicherung und Verarbeitung von Daten gemäß einem generischen Objektmodell

Die Erfindung betrifft ein System sowie ein Verfahren zur Strukturierung, Speicherung und Verarbeitung von Daten.

Üblicherweise werden Software-Applikationen unterschiedlichster Art zur Lösung technischer Aufgabenstellungen, z. B. dem Engineering eines Automatisierungssystem, eingesetzt, wobei jede dieser Software-Applikationen einerseits spezifische technische Aufgaben erfüllt, andererseits mit anderen Software-Applikationen zur Lösung einer technischen Aufgabe zusammenwirkt. Letzteres impliziert, dass die Software-Applikationen über Schnittstellen Daten austauschen. Die Schnittstellen, welche die einzelnen Software-Applikationen bieten, sowie die darüber transportierten Daten, sind meist sehr heterogen und proprietär. In der Regel werden die zu transportierenden Daten jeweils für den individuellen Austauschbedarf strukturiert. Über verschiedene Software-Applikationen hinweg führt das jedoch zu inkompatiblen Austauschstrukturen.

Der Erfindung liegt die Aufgabe zugrunde, den Datenaustausch zwischen verschiedenen Software-Applikationen zu vereinfachen.

Diese Aufgabe wird durch ein System zur Strukturierung, Speicherung und Verarbeitung von Daten gemäß einem generischen Objektmodell gelöst, wobei das Objektmodell mindestens ein erstes Element aufweist, welches einem Typ Objekt entspricht, wobei der Typ Objekt folgende Merkmale aufweist:

- eine eindeutige Bezeichnung der Identität des Objekts zur absoluten Referenzierung des Objekts,

- einen logischen Namen zur Benennung des Objekts und
- mindestens eine Verknüpfung mit einem zweiten Element, welches einem Typ Feature entspricht, wobei der Typ Feature folgende Merkmale aufweist:
  - einen im Bezug auf das jeweilige verknüpfte Objekt eindeutigen Namen und

die Möglichkeit der Verknüpfung mit weiteren Elementen vom Typ Objekt, mit weiteren Elementen vom Typ Feature und mit Daten.

Diese Aufgabe wird durch ein Verfahren zur Strukturierung, Speicherung und Verarbeitung von Daten gemäß einem generischen Objektmodell gelöst, wobei das Objektmodell mindestens ein erstes Element aufweist, welches einem Typ Objekt entspricht, wobei der Typ Objekt folgende Merkmale aufweist:

- eine eindeutige Bezeichnung der Identität des Objekts zur absoluten Referenzierung des Objekts,
- einen logischen Namen zur Benennung des Objekts und
- mindestens eine Verknüpfung mit einem zweiten Element, welches einem Typ Feature entspricht, wobei der Typ Feature folgende Merkmale aufweist:
  - einen im Bezug auf das jeweilige verknüpfte Objekt eindeutigen Namen und

die Möglichkeit der Verknüpfung mit weiteren Elementen vom Typ Objekt, mit weiteren Elementen vom Typ Feature und mit Daten.

Die Erfindung beruht auf der Idee, komplexe, vorzugsweise hierarchisch aufgebaute Datenmengen mit einem einheitlichen Objektmodell zu beschreiben und zu strukturieren. Alle Elemente des Typs Objekt haben die gleiche Grundstruktur, sind jedoch in unterschiedlichen Granularitätsstufen einsetzbar. Die Struktur eines übergeordneten Elements vom Typ Objekt findet sich also in der Struktur eines untergeordneten Elements vom Typ Objekt wieder. Das gesamte Objektmodell besitzt somit bis zur untersten Ebene eine

annähernd fraktale Struktur. Die Strukturierung der Datenmenge gelingt durch Replikation weniger Grundmuster und Grundstrukturen. Durch die Verwendung dieses Darstellungsprinzips (Objekt, Feature, etc.) kann erreicht werden, dass alle damit modellierten Datenbestände gemeinsame Grundstrukturen beinhalten, mit denen ein universelles Verständnis möglich ist. Alle Elemente stellen die Struktur-Information eines Datenbestands dar. Applikationen können so auf einheitliche Weise auf die Daten zugreifen bzw. in den Objektgeflechten navigieren. Ferner können beliebige, heute noch nicht bekannte Abbildungsanforderungen erfüllt werden, die dann auch wieder in dieses grundsätzliche Verständnis der Einheitlichkeit einfließen und von anderen Applikationen verstanden werden. Applikationen die sich diesem einheitlichen Format zukünftig anpassen, genießen dann automatisch auch die Kompatibilität mit allen vorherigen.

Die Bezeichnung der Identität eines Objektes wird nach der Erzeugung nie mehr geändert, insbesondere bleibt sie bestehen beim Verschieben des Objekts innerhalb eines Datenbestandes oder dem Einfügen des Objektes in andere Datenbestände. Die Identität dient zur eindeutigen Identifizierung eines Objekts, d. h. über die Identität kann ein Objekt absolut, also ohne Bezug zu seiner Umwelt bzw. seinem Kontext, referenziert werden.

Neben einer Identität hat jedes Objekt einen logischen Namen. Der Name kann im Gegensatz zur Identität geändert werden und muss auch nicht global eindeutig sein. Wenn jedoch Eindeutigkeit unter den Namen der Objekte in jedem Feature herrscht, dann können diese zur Bildung sogenannter Pfad-Referenzen (Referenzen eines Objekts im Bezug auf seine Umwelt) dienen.

Elemente des Typs Feature bilden die Substruktur der Objekte. Sie gruppieren z. B. Parameter, Referenzen, Subobjekte,

Connectoren und Connections des Objekts und können auch selbst wieder über Features strukturiert werden.

Gemäß einer vorteilhaften Ausgestaltung der Erfindung weist der Typ Objekt als weitere Merkmale eine Kennzeichnung des Objektstyps und eine Kennzeichnung einer Version des Objekts auf. Dies ist insbesondere vorteilhaft zur Strukturierung von komplexen, zeitlich variablen Datenbeständen.

Eine weiter verbesserte Strukturierung der Daten lässt sich erreichen, wenn die durch ein Element vom Typ Feature verknüpften und gruppierten Elemente eine logisch zusammengehörige Teilmenge aller Elemente eines Objekts bilden. Grundlage der Gruppierung können zum einen eine logische Zusammengehörigkeit der Elemente des Objekts zu einer bestimmten "Sicht" (z. B. HMI, Hardware, Software) auf das Objekt sein. Mit dieser Unterteilung können die jeweiligen Applikationen leichter jene Objekt-Daten lesen, die sie interessieren. Zum anderen können Features zur Erweiterung von bestehenden Objekten um spezifische weitere Objektinformationen verwendet werden, die zum Objekt hinzugefügt werden sollen und evtl. nur für bestimmte Applikationen von Interesse sind. Dieser Weg kann sinnvollerweise im Gegensatz zur Erweiterung durch Ableitung gewählt werden, um Produkte, die mit bestehenden Typen arbeiten, nicht inkompatibel werden zu lassen. Durch die Erweiterung über neue Features muss auf bestehende Applikationen keine Rücksicht genommen werden.

Des Weiteren können die Objekte über Features auch weitere (Sub-)Objekte und Referenzen zu anderen Objekten enthalten. Über die Aggregation entsteht so ein Baum aus Objekten, wobei durch die Referenzen Querbezüge zwischen den Elementen dieses Baums dargestellt werden können. Vorteilhafterweise werden keine Rollen von Objekten explizit spezifiziert. Die Rollen werden implizit durch die Position eines Objekts im

Verhältnis zu anderen Objekten dargestellt, beziehungsweise durch die Referenzen von und zu anderen Objekten ausgedrückt.

Wird das Objektmodell durch eine erweiterbare Kennzeichnungssprache beschrieben (z. B. XML = Extensible Markup Language), so erreicht man neben Einheitlichkeit und Erweiterbarkeit auch systematische Validierbarkeit.

Üblicherweise bilden Datenbestände, welche beim Engineering von Automatisierungssystemen Verwendung finden, umfangreiche und komplexe hierarchische Strukturen. Um deren strukturellen Inhalt einheitlich und transparent für verschiedene beteiligte Applikationen zur Verfügung zu stellen, wird die Verwendung des erfindungsgemäßen Systems bzw. Verfahrens zum Engineering einer Automatisierungslösung vorgeschlagen.

Nachfolgend wird die Erfindung anhand der in den Figuren dargestellten Ausführungsbeispiele näher beschrieben und erläutert.

Es zeigen:

FIG 1 die Grundidee des Objektmodells in Form eines UML-Diagramms und

FIG 2 ein System zum Engineering einer Automatisierungslösung.

In FIG 1 wird die Grundidee des Objektmodells 10 in Form eines UML-Diagramm dargestellt. UML (= Unified Modeling Language) ist eine durch die Object Management Group (OMG) standardisierte graphische Sprache zur Beschreibung objektorientierter Modelle. Im Mittelpunkt des Objektmodells 10 steht der Typ Objekt 100. Im Ausführungsbeispiel besitzt jedes Objekt 100 die Attribute ID 2, OType 5, Version 4 und Name 3. Die ID 2 ist hierbei eine eindeutige Bezeichnung, die sich nie ändert. Die ID 2 kann zum Beispiel eine GUID (=

Globally Unique Identifier) sein. Sie dient zur eindeutigen Identifizierung des Objektes 100, d. h. über die ID 2 kann das Objekt 100 absolut, also ohne Bezug zu seiner Umwelt bzw. seinem Kontext, referenziert werden. Jedem Objekt 100 wird ein Name 3 zugeordnet. Über den Namen 3 kann das Objekt 100 ebenfalls referenziert werden.

Wie im Diagramm von FIG 1 zu sehen, bilden Features 20 die Substruktur der Objekte 100. Sie gruppieren die Parameter 30, Referenzen 60, Sub-Objekte 100, Connectoren 40 und Connections 50 des Objekts 100 und können auch selbst wieder über Features 20 strukturiert werden. Die Verknüpfung zum SubObjekt 100 ist im UML-Diagramm von FIG 1 mit dem Bezugszeichen 70 gekennzeichnet, das SubObjekt 100 erhält jedoch das gleiche Bezugszeichen wie das oben erwähnte Objekt 100, da es die gleiche Struktur aufweist. Grundlage der Gruppierung sind zum einen die logische Zusammengehörigkeit der Bestandteile des Objekts 100 zu einer bestimmten "Sicht" (z.B. HMI, Hardware, Software) auf das Objekt 100. Mit dieser Unterteilung können die jeweiligen Tools leichter jene Objekt-Daten lesen, die sie interessieren.

Zum anderen bilden Features 20 die Einheit der Erweiterung von Objekten 100 um produktspezifische Bestandteile. Features 20 können somit zur Erweiterung von bestehenden Objekttypen um spezifische weitere Objektinformationen verwendet werden, die zum Objekt 100 hinzugefügt werden sollen und evtl. nur für bestimmte Applikationen von Interesse sind. Dieser Weg wurde im Gegensatz zur Ableitung gewählt, um Produkte, die mit bestehenden Typen arbeiten, nicht inkompatibel werden zu lassen. Wenn ein Objekttyp um Daten eines anderen Produkts erweitert werden soll, so wird ein neues Feature 20 definiert, das dann zu dem bestehenden Objekt 100 dazugefügt wird. Die Definition des ursprünglichen Objekttyps bleibt dabei bestehen, damit die Tools, die mit dem bisherigen Objekttyp arbeiten, nicht beeinträchtigt werden. Durch die Erweiterung über Features 20 muss auf bestehende

Applikationen keine Rücksicht genommen werden. Features 20, die einen im Bezug auf das jeweilige Objekt 100 eindeutigen Namen 21 tragen, haben jeweils 1 zu n - Beziehungen zu Parametern 30, Connectoren 40 und Connections 50. Des Weiteren können die Objekte 100 über Features 20 auch Sub-Objekte 100 aggregieren und Referenzen 60/Relationen zu anderen Objekten 100 enthalten. Über die Aggregation entsteht ein Baum aus Objekten 100. Durch die Referenzen 60 können Querbezüge zwischen den Elementen dieses Baums dargestellt werden. Die Parameter 30, Connectoren 40 und Connections 50 stellen bildlich gesprochen das Laub des Baumes, also im Bezug auf die zu modellierenden Datenbestände die eigentlichen Daten dar. Features 20 können selbst wieder Features 20 enthalten. Objekte, Features und Referenzen stellen die Struktur-Information eines Datenbestandes dar.

Die Identität ID 2 eines Objektes 100 wird nach der Erzeugung nie mehr geändert. Insbesondere bleibt sie bestehen beim Verschieben des Objekts 100 innerhalb eines Datenbestandes und beim Einfügen des Objektes 100 in andere Bestände. Die ID 2 dient als absolute Referenz auf ein Objekt 100. D. h. über die ID 2 kann ein Objekt 100 absolut, also ohne Bezug zu seiner Umwelt/zu seinem Kontext referenziert werden. Neben einer ID 2 hat jedes Objekt 100 einen (logischen) Namen 3. Der Name 3 kann im Gegensatz zur ID 2 geändert werden und muss auch nicht global eindeutig sein. Wenn jedoch Eindeutigkeit unter den Namen 2 der SubObjekte 100 in jedem Feature 20 herrscht, dann können diese zur Bildung sogenannter Pfad-Referenzen (Referenzen, die ein Objekt 100 im Bezug auf seine Umwelt referenzieren) dienen.

Es werden keine Rollen von Objekten 100 explizit spezifiziert. Die Rollen werden vielmehr implizit durch die Position eines Objekts 100 im Verhältnis zu anderen Objekten 100 dargestellt, beziehungsweise sie werden durch die Referenzen 60 von und zu anderen Objekten 100 ausgedrückt.

Durch die Verwendung dieses Darstellungsprinzips (Objekt 100, Feature 20, etc.) kann erreicht werden, dass alle damit modellierten Datenbestände gemeinsame Grundstrukturen beinhalten, mit denen ein universelles Verständnis möglich ist, sprich Applikationen auf einheitliche Weise auf die Inhalte zugreifen bzw. in den Objektgeflechten navigieren können. Ferner können beliebige, heute noch nicht bekannte Abbildungsanforderungen erfüllt werden, die dann auch wieder in dieses grundsätzliche Verständnis der Einheitlichkeit einfließen, und von anderen Applikationen verstanden werden. Applikationen die sich diesem einheitlichen Format zukünftig anpassen, genießen dann automatisch auch die Kompatibilität mit allen vorherigen.

Wenn man diesen Gedanken in Schemas der Metasprache XML (=Extensible Markup Language) niederlegt, so erreicht man neben Einheitlichkeit und Erweiterbarkeit auch systematische Validierbarkeit. Das obige Objektmodell 10 sei im Folgenden anhand einer Darstellung als Schema in XML gezeigt:

```
<xsd:complexType name="ObjectT">
      <xsd:sequence>
         <xsd:element name="Features" type="FeaturesT" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="ID" type="IdT" use="required"/>
      <xsd:attribute name="Name" type="xsd:string" use="required"/>
      <xsd:attribute name="Version" type="VersionT" use="optional"/>
      <xsd:attribute name="Type" type="xsd:QName" use="optional"/>
</xsd:complexType>

<xsd:complexType name="FeatureT" />
     <xsd:complexContent>
           <xsd:sequence>
                 <xsd:element ref="Parameter" minOccurs="0"/>
                 <xsd:element ref="Reference" minOccurs="0" />
                 <xsd:element ref="Object" minOccurs="0" />
           </xsd:sequence>
```

```
            <xsd:attribute name="Name" type="xsd:string" use="required"/>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="ParameterT">
        <xsd:annotation>
            <xsd:documentation>Base type for all DIA-X parameters that
            are used within features of objects</xsd:documentation>
        </xsd:annotation>
        <xsd:attribute name="MustUnderstand" type="xsd:boolean"
        use="optional" default="false" />
        <xsd:attribute name="Name" type="xsd:string" use="optional" />
    </xsd:complexType>
```

Die der Erfindung zugrundeliegende Idee wird anhand eines weiteren Ausführungsbeispiels näher erläutert. Darzustellen sei ein Symbol, wie diese z. B. in der Automatisierungstechnik üblich sind. So ein Symbol enthält neben seinem Namen noch einen Typ, eine Richtung und einen Wert. Das zu zeigende Beispiel-Symbol sei dieses:

S7_AO_Niveau E0.3

Wobei "S7_AO_Niveau" der Name des Symbols ist. Typ und Richtung sind zusammen mit dem Wert in der Bezeichnung E0.3 in der Weise verschlüsselt, das "E" die (deutschsprachige) Richtungsbezeichnung für "Eingang" bedeutet, und in der Punkt-Darstellung sich die Adressierung eines Bits innerhalb eines Wortes ausdrückt. Das Beispiel-Symbol würde gemäß dem obigen Objektmodell 10 wie folgt dargestellt werden, und auch validierbar sein, wenn man das oben gezeigte allgemeine Schema noch mit den anschließend gezeigten symbolspezifischen Verfeinerungen versieht. Eine Instanz des Beispiel-Symbols "S7_AO_Niveau" ist als XML-Schema folgendermaßen definiert:

```
<base:Symbol ID="{5ED19706-3840-4da0-ADD2-
    27491C0A58BB}"Name="S7_AO_Niveau">
    <base:AddressFeature>
      <base:SymbolAddress Direction="In" AddressType="Bit" Value="3.0" />
    </base:AddressFeature>
</base:Symbol>
```

Im Folgenden werden die genannten symbol-spezifischen Verfeinerungen beschrieben, mit deren Hilfe ein so beschriebenes Symbol validierbar wird. Als Erstes ist vom allgemeinen Typ "Objekt" ein symbol-spezifischer Objekttyp (hier genannt "SymbolT") abzuleiten. Dieser enthält wie oben erklärt auch ein Feature (da er ja abgeleitet ist), nämlich wiederum ein symbol-spezifisches Feature. Es sei "SymbolAdressFeatureT" genannt.

```
<xsd:element name="Symbol" type="SymbolT"
    substitutionGroup="diax:Object"/>

<xsd:complexType name="SymbolT">
      <xsd:complexContent>
            <xsd:extension base="diax:ObjectT">
                  <xsd:sequence>
                        <xsd:element name="AddressFeature"
    type="SymbolAddressFeatureT"/>
                  </xsd:sequence>
            </xsd:extension>
      </xsd:complexContent>
</xsd:complexType>
```

Dieses symbol-spezifische Feature (genannt "SymbolAddressFeatureT" - wobei "T" für Typ steht) enthält gemäß obigem Basis-Objekt einen Parameter, nämlich wiederum einen symbol-spezifischen, der "SymbolAddressT" heißt:

```
Feature SymbolAddressFeatureT
```

```
<xsd:complexType name="SymbolAddressFeatureT">
      <xsd:complexContent>
            <xsd:extension base="diax:FeatureT">
                  <xsd:sequence>
                        <xsd:element name="SymbolAddress"
    type="SymbolAddressT"/>
                  </xsd:sequence>
            </xsd:extension>
      </xsd:complexContent>
</xsd:complexType>
```

Der symbol-spezifische Parameter "SymbolAddressT" wird im Folgenden definiert. Er enthält die restlichen Informationen: Datentyp, Richtung, Wert.

```
Parameter SymbolAddressT

<xsd:complexType name="SymbolAddressT">
    <xsd:complexContent>
      <xsd:extension base="diax:ParameterT">
            <xsd:attribute name="AddressType" type="AddressTypeEnumT"
    use="required"/>
            <xsd:attribute name="Direction" type="DirectionEnumT"
    use="required"/>
            <xsd:attribute name="Value" type="xsd:string"
    use="required"/>
      </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Mit weiteren Spezifikationen sind die im Adress-Parameter verwendeten Attribute ‚Typ' und ‚Richtung' definiert. Der "Wert" ist letztlich nur ein xsd:string ohne Einschränkungen. Damit genügt das obige Beispiel-Symbol dem generischen Grund-Objektmodell 10 und ist voll validierbar festgelegt.

Üblicherweise sind Datenbestände 210, wie sie im Engineering 220 von Automatisierungssystemen 230 vorkommen, als umfangreiche, komplexe hierarchische Strukturen beschaffen. Um deren strukturellen Inhalt einheitlich, und transparent für andere zu machen, kann ein einfaches erfindungsgemäßes Objektmodell 10 als zentrales, generisches Grundelement der Darstellung definiert werden. Das sei im Folgenden am Beispiel eines Hardware-Projekts 200 mit seinem strukturellen Aufbau demonstriert (siehe FIG 2). Das mit "Projekt" 200 benannte hierarchische Gefüge beinhalte eine Verarbeitungs-station 201, welche als "S7 300" bezeichnet wird. Diese enthalte auf einem "Rack UR" 202 eine "CPU 315" 203, die unter vielem anderen in ihrem Symbol-Container das Symbol "S7_AO_Niveau" enthält. Zur validierbaren Darstellung dieser Strukturen sind natürlich wiederum spezifische Verfeinerungen der Standard Schemas notwendig (zum Beispiel das StructuralFeature, das den Aufbau eines Objektes beschreibt), auf deren Darstellung hier jedoch verzichtet wird. Hier sei der Hinweis genug, dass derer beliebig viele durch entsprechende Ableitung erstellt werden können, wodurch alle dargestellten Daten dann auch systematisch validierbar sind.

```
<base:Project ID="{3E397603-9E8C-46EC-8B41-10A60FAA3B17}" Name="Project">
 <base:StructuralFeature>
  <base:Device ID="{EEAD7EA6-2F73-46D8-BCF2-2DAC712CF813}" Name="S7300">
   <base:StructuralFeature>
    <base:Device ID="{E378890F-DEA9-41EF-8C35-6EEF76FD748B}" Name="UR">
     <base:StructuralFeature>
      <base:Device ID="{85852272-12E2-4D4...}" Name="CPU315">
       <base:SoftwareFeature>
        <base:Symbol ID="{85F306C6-412…}" Name="S7_AO_Niveau">
         <base:AddressFeature>
          <base:SymbolAddress Direction="In" AddressType="Bit"
Value="0.3"/>
         </base:AddressFeature>
        </base:Symbol>
...
```

Zusammenfassend betrifft die Erfindung somit ein System sowie ein Verfahren zur Strukturierung, Speicherung und Verarbeitung von Daten. Der Datenaustausch zwischen verschiedenen Software-Applikationen wird vereinfacht durch die Strukturierung, Speicherung und Verarbeitung gemäß einem generischen Objektmodell 10, wobei das Objektmodell 10 mindestens ein erstes Element aufweist, welches einem Typ Objekt 100 entspricht, wobei der Typ Objekt 100 folgende Merkmale aufweist:

- eine eindeutige Bezeichnung 2 der Identität des Objekts 100 zur absoluten Referenzierung des Objekts 100,
- einen logischen Namen 3 zur Benennung des Objekts 100 und
- mindestens eine Verknüpfung 6 mit einem zweiten Element, welches einem Typ Feature 20 entspricht, wobei der Typ Feature 20 folgende Merkmale aufweist:
  - einen im Bezug auf das jeweilige verknüpfte Objekt 100 eindeutigen Namen 21 und
  - die Möglichkeit der Verknüpfung mit weiteren Elementen vom Typ Objekt 100, mit weiteren Elementen vom Typ Feature 20 und mit Daten 30, 40, 50.

Patentansprüche

1. System zur Strukturierung, Speicherung und Verarbeitung von Daten gemäß einem generischen Objektmodell (10), wobei das Objektmodell (10) mindestens ein erstes Element aufweist, welches einem Typ Objekt (100) entspricht, wobei der Typ Objekt (100) folgende Merkmale aufweist:
- eine eindeutige Bezeichnung (2) der Identität des Objekts (100) zur absoluten Referenzierung des Objekts (100),
- einen logischen Namen (3) zur Benennung des Objekts (100) und
- mindestens eine Verknüpfung (6) mit einem zweiten Element, welches einem Typ Feature (20) entspricht, wobei der Typ Feature (20) folgende Merkmale aufweist:
  - einen im Bezug auf das jeweilige verknüpfte Objekt (100) eindeutigen Namen (21) und
  - die Möglichkeit der Verknüpfung mit weiteren Elementen vom Typ Objekt (100), mit weiteren Elementen vom Typ Feature (20) und mit Daten (30, 40, 50).

2. System nach Anspruch 1,
dadurch gekennzeichnet,
dass der Typ Objekt (100) als weitere Merkmale eine Kennzeichnung des Objektstyps (5) und eine Kennzeichnung einer Version (4) des Objekts (100) aufweist.

3. System nach einem der vorhergehenden Ansprüche,
dadurch gekennzeichnet,
dass die durch ein Element vom Typ Feature (20) verknüpften Elemente eine logisch zusammengehörige Teilmenge aller Elemente eines Objekts (100) bilden.

4. System nach einem der vorhergehenden Ansprüche,
dadurch gekennzeichnet,
dass die Elemente des Objekts (100) durch Referenzen (60) verknüpft sind.

5. System nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, dass das Objektmodell (10) durch eine erweiterbare Kennzeichnungssprache beschrieben wird.

6. Verfahren zur Strukturierung, Speicherung und Verarbeitung von Daten gemäß einem generischen Objektmodell (10), wobei das Objektmodell (10) mindestens ein erstes Element aufweist, welches einem Typ Objekt (100) entspricht, wobei der Typ Objekt (100) folgende Merkmale aufweist:

- eine eindeutige Bezeichnung (2) der Identität des Objekts (100) zur absoluten Referenzierung des Objekts (100),
- einen logischen Namen (3) zur Benennung des Objekts (100) und
- mindestens eine Verknüpfung (6) mit einem zweiten Element, welches einem Typ Feature (20) entspricht, wobei der Typ Feature (20) folgende Merkmale aufweist:
  - einen im Bezug auf das jeweilige verknüpfte Objekt (100) eindeutigen Namen (21) und
  - die Möglichkeit der Verknüpfung mit weiteren Elementen vom Typ Objekt (100), mit weiteren Elementen vom Typ Feature (20) und mit Daten (30, 40, 50).

7. Verwendung eines Systems bzw. eines Verfahrens nach einem der vorhergehenden Ansprüche zum Engineering (220) eines Automatisierungssystems (230).

Zusammenfassung

Strukturierung, Speicherung und Verarbeitung von Daten gemäß einem generischen Objektmodell

Die Erfindung betrifft ein System sowie ein Verfahren zur Strukturierung, Speicherung und Verarbeitung von Daten. Der Datenaustausch zwischen verschiedenen Software-Applikationen wird vereinfacht durch die Strukturierung, Speicherung und Verarbeitung gemäß einem generischen Objektmodell (10), wobei das Objektmodell (10) mindestens ein erstes Element aufweist, welches einem Typ Objekt (100) entspricht, wobei der Typ Objekt (100) folgende Merkmale aufweist:

- eine eindeutige Bezeichnung (2) der Identität des Objekts (100) zur absoluten Referenzierung des Objekts (100),
- einen logischen Namen (3) zur Benennung des Objekts (100) und
- mindestens eine Verknüpfung (6) mit einem zweiten Element, welches einem Typ Feature (20) entspricht, wobei der Typ Feature (20) folgende Merkmale aufweist:
  - einen im Bezug auf das jeweilige verknüpfte Objekt (100) eindeutigen Namen (21) und
  - die Möglichkeit der Verknüpfung mit weiteren Elementen vom Typ Objekt (100), mit weiteren Elementen vom Typ Feature (20) und mit Daten (30, 40, 50).

FIG 1

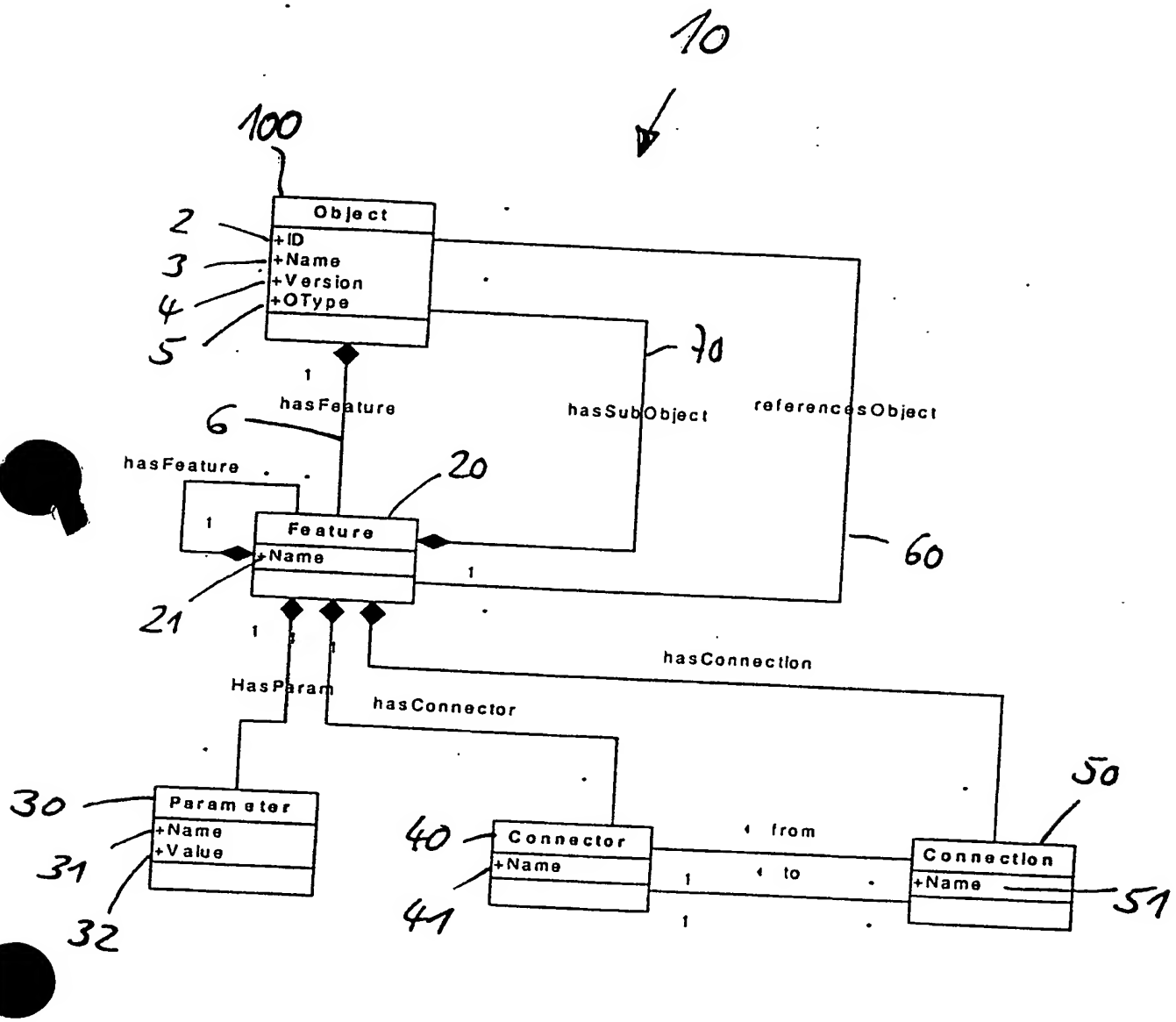
10
100
Object
+ID
+Name
+Version
+OType
2
3
4
5
70
1
hasFeature
6
hasSubObject
referencesObject
hasFeature
20
Feature
+Name
60
21
1
hasConnection
HasParam
hasConnector
30
Parameter
+Name
+Value
31
32
40
Connector
+Name
41
from
to
1
50
Connection
+Name
51

FIG 1

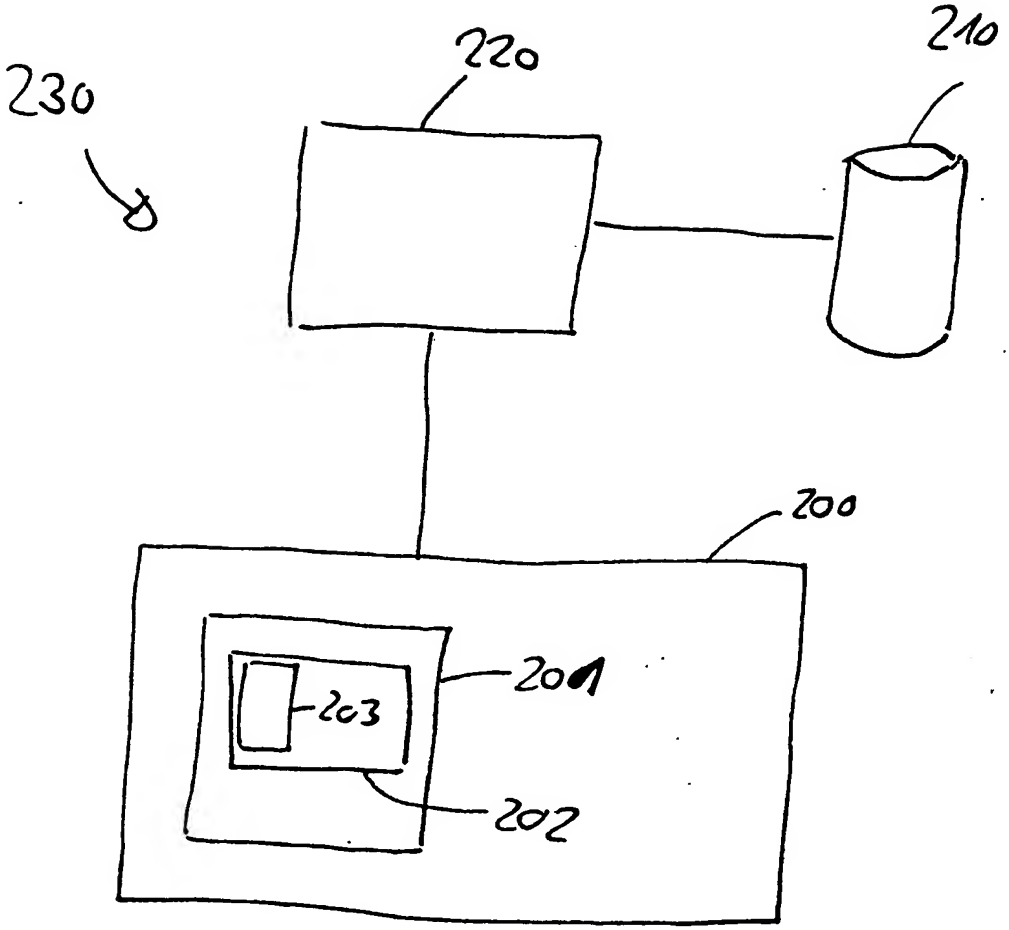

FIG 2